In the Claims:

Please amend claim 17 as follows:

1. (Previously Presented) A method of facilitating optimization processing in a compiler, comprising the steps of:

(a) storing, in a language-specific-rule table, assignment rules which are specified for at least two programming languages allowing vector representation of each of said programming languages;

(b) analyzing a program code which includes one or more instructions, and is described in a selected one of said programming languages, based on said assignment rules, to obtain an analysis result; and

(c) embedding said analysis result in said program code;

wherein in said step (a), said assignment rules are stored in said language-specific-rule table as one or more language-specific-information analyzing functions, and

said step (b) comprises the substeps of,

(d) reading out, from said language-specific-rule table, at least one of said one or more language-specific-information analyzing functions which is needed for analyzing said program code, and

(e) determining values of or relationships between variables included in said program code, based on said at least one of said one or more language-specific-information analyzing functions read out in said step (d), and producing said analysis result which includes the determined values of or relationships between the variables;

wherein said step (b) further comprises the substep of,

(f) said at least one of said one or more language-specific-information analyzing functions read out in said step (d) is registered in a check function table for use in said step (e).

2-3. (Cancelled)

4. (Previously presented) A method according to claim 1, wherein the operation in step (b) is performed for each instruction set which is comprised of at least one of said one or more instructions, and

in said step (c), said analysis result is embedded in a position preceding said each instruction set in said program code.

5. (Original) A method according to claim 1, wherein said program code is a source code.

6. (Original) A method according to claim 1, wherein said program code is an intermediate code.

7. (Previously Presented) An apparatus for facilitating optimization processing in a compiler, comprising:

a language-specific-rule table which stores assignment rules which are specified for at least two programming languages allowing vector representation of each of said programming languages;

an analyzing unit which analyzes a program code which includes one or more instructions, and is described in a selected one of said programming languages, based on said assignment rules, to obtain an analysis result; and

an embedding unit which embeds said analysis result in said program code;

wherein said language-specific-rule table stores said assignment rules as one or more language-specific-information analyzing functions, and

said analyzing unit comprises,

a readout unit which reads out, from said language-specific-rule table, at least one of said one or more language-specific-information analyzing functions which is needed for analyzing said program code, and

a determination unit which determines values of or relationships between variables included in said program code, based on said at least one of said one or more language-specific-information analyzing functions read out by said readout unit, and produces said analysis result which includes the determined values of or relationships between the variables;

wherein said analyzing unit comprises,

a check function table in which said at least one of said one or more language-specific-information analyzing functions read out by said readout unit is registered for use by said determination unit.

8-9. (Cancelled)

10. (Previously presented) An apparatus according to claim 7, wherein the operation of said analyzing unit is performed for each instruction set which is comprised of at least one of said one or more instructions, and

said embedding unit embeds said analysis result in a position preceding said each instruction set in said program code.

11. (Original) An apparatus according to claim 7, wherein said program code is a source code.

12. (Original) An apparatus according to claim 7, wherein said program code is an intermediate code.

13. (Previously Presented) A product for use with an apparatus for facilitating optimization processing in a compiler, said product, when used with said apparatus, is able to output control information which directs the apparatus to comprise:

a language-specific-rule table which stores assignment rules which are specified for at least two programming languages allowing vector representation of each of said programming languages;

an analyzing unit which analyzes a program code which includes one or more instructions, and is described in a selected one of said programming languages, based on said assignment rules, to obtain an analysis result; and

an embedding unit which embeds said analysis result in said program code;

wherein said language-specific-rule table stores said assignment rules as one or more language-specific-information analyzing functions, and

said analyzing unit comprises,

a readout unit which reads out, from said language-specific-rule table, at least one of said one or more language-specific-information analyzing functions which is needed for analyzing said program code, and

a determination unit which determines values of or relationships between variables included in said program code, based on said at least one of said one or more language-specific-information analyzing functions read out by said readout unit, and produces said analysis result which includes the determined values of or relationships between the variables.

14. (Cancelled)

15. (Original) A product according to claim 13, wherein said program code is a source code.

16. (Original) A product according to claim 13, wherein said program code is an intermediate code.

17. (Currently Amended) A compiler comprising:

a front end unit which performs syntax analysis of a source code which is described in one of at least two predetermined programming languages allowing vector representation of each of said predetermined programming languages, to produce an intermediate code;

a language-specific-rule table which stores assignment rules which are specified for said predetermined programming languages;

an analyzing unit which analyzes said intermediate code which includes one or more instructions and is described in a selected one of said predetermined programming languages, based on said one or more assignment rules, to obtain an analysis result;

an embedding unit which embeds said analysis result in said ~~program~~ intermediate code to produce a modified intermediate code; and

an optimizing unit which performs an operation of optimizing said modified intermediate code;

wherein said language-specific-rule table stores said assignment rules as one or more language-specific-information analyzing functions, and

said analyzing unit comprises,

a readout unit which reads out, from said language-specific-rule table, at least one of said one or more language-specific-information analyzing functions which is needed for analyzing said intermediate code, and

a determination unit which determines values of or relationships between variables included in said intermediate code, based on said at least one of said one or more language-specific-information analyzing functions read out by said readout unit, and produces said analysis result which includes the determined values of or relationships between the variables.

18. (Cancelled)

19. (Previously Presented) A compiler comprising:

a language-specific-rule table which stores assignment rules which are specified for at least two programming languages allowing vector representation of each of said programming languages;

an analyzing unit which analyzes a source code which includes one or more instructions, and is described in a selected one of said programming languages, based on said assignment rules, to obtain an analysis result;

an embedding unit which embeds said analysis result in said source code to produce a modified source code;

a syntax analyzing unit which performs syntax analysis of said modified source code which is described in one of said programming languages, to produce an intermediate code; and

an optimizing unit which performs an optimization operation on said intermediate code;

wherein said language-specific-rule table stores said assignment rules as one or more language-specific-information analyzing functions, and

said analyzing unit comprises,

a readout unit which reads out, from said language-specific-rule table, at least one of said one or more language-specific information analyzing functions which is needed for analyzing said source code, and

a determination unit which determines values of or relationships between variable included in said source code, based on said at least one of said one or more language-specific-information analyzing functions read out by said readout unit, and produces said analysis result which includes the determined values of or relationships between the variables.

20. (Cancelled)